Факультет Математики и Информатики  
Департамент Информатики

**Лабораторная работа №2**

По курсу “Алгоритмика графов”

Тема:”Устойчивые множества в неориентированных графах.

Алгоритм Беднарека и Толби.”

Выполнил студент группы I1902:  
Чобану Артём

Проверил:

Цуркану Кэлин

Кишинэу, 2020

Задачи 2 и 19 (n и 21-n, где n – порядковый номер)

**2. Каково число диагоналей в выпуклом многоугольнике с n вершинами.**

Решение:

1. От любой вершины **невозможно** провести диагональ к самой **себе**, и к **соседним** вершинам. Следовательно, количество диагоналей, которые можно провести от каждой вершины равно **n - 3**.
2. Вычислим число диагоналей, которые можно провести от всех вершин:
3. В предыдущем пункте мы посчитали каждую диагональ два раза, так как мы считали все проведённые диагонали из каждой вершины, но не учитывали, что половина из них совпадают.

Следовательно, поделим формулу на 2 и получим число диагоналей в выпуклом многоугольнике с n вершинами:

Ответ:

**19. Для неориентированного графа G имеет ли место неравенство**

**?**

– это число рёберного покрытия графа, т.е. число рёбер в наименьшем рёберном покрытии графа.

– число внутри устойчивости, т.е. число вершин в наибольшем внутри устойчивом множестве.

**Решение:**

Исходя из условия, каждое ребро из инцидентно хотя бы одной вершине графа из множества .

Так как – это внутри устойчивое множество, то на каждую вершину из приходится хотя бы одно ребро из .

Следовательно,

Реализация алгоритма Беднарека-Толби на языке программирования C#:

Кроме того, проект можно получить по ссылке: <https://github.com/ArtiomCiobanu/GraphsTest>

public void BednarekTaulbeeAlgorithm()  
{  
 int length = Vertices.Count;  
 List<List<Vertex>> L = new List<List<Vertex>>()  
 {  
 new List<Vertex>()  
 {  
 Vertices.First()  
 }  
 };  
 for (int k = 1; k < length; k++)  
 {  
 //Шаг 1  
 var Xk = Vertices.Take(k - 1).ToList(); //Множество Xk  
 var currentX = Vertices[k]; //Xk+1  
 //Шаг 2  
 var Y = Xk.Where(v => !currentX.IsAdjacentTo(v)).ToList();  
 //Y.Add(currentX); //Для алгоритма необязательно  
 //Шаг 3  
 List<List<Vertex>> I = L.Select(l => l.Intersect(Y).ToList())  
 .Distinct().ToList();  
 //Шаг 4  
 var maxISets = GraphTools.GetMaxSets(I);  
 //Шаг 5   
 var currentL = new List<List<Vertex>>();  
 foreach (var M in L)  
 {  
 var mIntersectY = M.Intersect(Y).ToList();  
  
 if (mIntersectY.Count == M.Count)  
 {  
 M.Add(currentX);  
 }  
 else if (I.Contains(mIntersectY))  
 {  
 mIntersectY.Add(currentX);  
 }  
 currentL.Add(M);  
 }  
 //Шаг 6  
 L = currentL.ToList();  
 var p = L.Select(m => m.Select(v => v.Index).ToArray());  
 Tools.PrintIntMatrix(p);  
 }  
}

Метод **BednarekTaulbeeAlgorithm** находится в классе Graph, использованный так же и в Лабораторной работе No.1

Вызов метода:

Console.WriteLine();  
Graph graph1 = new Graph(adjacencyMatrix);  
graph1.BednarekTaulbeeAlgorithm();

Код всего метода **Main**:

private static void Main()  
{  
 string input = File.ReadAllText(@"Input.txt");  
 var adjacencyMatrix = Tools.GetIntArrayFromString(input);  
  
 Console.WriteLine("Матрица смежности:");  
 Tools.PrintIntMatrix(adjacencyMatrix);  
  
 Console.WriteLine();  
 Graph graph1 = new Graph(adjacencyMatrix);  
 graph1.BednarekTaulbeeAlgorithm();  
}

Код метода **GetMaxSets**, возвращающий только максимальные множества:

public static List<List<Vertex>> GetMaxSets(List<List<Vertex>> input)  
{  
 for (int i = 0; i < input.Count(); i++)  
 {  
 var current = input[i];  
 for (int j = 0; j < input.Count; j++)  
 {  
 if (current.Intersect(input[j]).Count() > current.Count())  
 {  
 input.Remove(current);  
 }  
 }  
 }  
  
 return input;  
}